Solving Survo puzzles using matrix combinatorial products

Vehkalahti, Kimmo

2015

Vehkalahti , K & Sund , R 2015 , ' Solving Survo puzzles using matrix combinatorial products
' , Journal of Statistical Computation and Simulation , vol. 85 , pp. 2666-2681 . https://doi.org/10.1080/00949655.201

http://hdl.handle.net/10138/232431
https://doi.org/10.1080/00949655.2014.899363

acceptedVersion

To appear in the *Journal of Statistical Computation and Simulation*
Vol. 00, No. 00, Month 20XX, 1–16

Solving Survo Puzzles using Matrix Combinatorial Products [SPECIAL ISSUE: Matrix]

Kimmo Vehkalahti* and Reijo Sund

University of Helsinki, Helsinki, Finland

(*10 January 2014*)

We investigate combinatorial matrix problems that are related to restricted integer partitions. They arise from *Survo puzzles*, where the basic task is to fill an $m \times n$ table by integers $1, 2, \ldots, mn$, so that each number appears only once, when the column sums and the row sums are fixed. We present a new computational method for solving Survo puzzles with binary matrices that are recoded and combined using the Hadamard, Kronecker, and Khatri–Rao products. The idea of our method is based on using the matrix interpreter and other data analytic tools of Survo R, which represents the newest generation of the Survo computing environment, recently implemented as a multiplatform, open source R package. We illustrate our method with detailed examples.

Keywords: Survo puzzle; integer partitions; matrix combinatorial products; Survo R; Magic square

AMS Subject Classification: 05A17; 15B34; 15B36; 15A99; 68N01; 97R80

1. Introduction

Survo puzzle is a challenging, numerical puzzle that has encouraged both recreation and research since its invention by Mustonen [1] in 2006. The primary task is simply to fill an $m \times n$ table by integers $1, 2, \ldots, mn$, so that each number appears only once, when both the column sums and the row sums are fixed. Some of the mn integers may also be readily given, but in an *open* Survo puzzle, they are all missing. The level of difficulty of a Survo puzzle varies, depending, e.g., on the size of the table and the number of the missing integers [2]. Finding solutions for any non-trivial Survo puzzle is a combinatorial problem, where a crucial role is played by restricted integer partitions with m (or n) distinct parts. This is a common type of restricted partitions [3, 4].

There is an interesting connection between Survo puzzles and Magic squares [5, 6]. The original motivation behind the Survo puzzles was to create a new cross sum puzzle to be played on an open rectangular grid with simpler rules compared to somewhat related puzzles, such as Kakuro [1]. However, when $m = n$ and all the row and column sums are equal to the same number, a Survo puzzle resembles a Magic square. We will highlight this connection in one of the examples.

Various methods and algorithms of solving Survo puzzles have been suggested [1, 2]. In this paper, we present a new, computational method for solving Survo puzzles with binary matrices that are recoded and combined using the Hadamard, Kronecker, and Khatri–Rao products [7]. To accomplish this task, we employ the matrix interpreter and other data analytic tools of the Survo computing environment, which is the lifework of professor Seppo Mustonen since the early 1960s [8, 9]. Survo binds together a selection of

*Corresponding author. Email: kimmo.vehkalahti@helsinki.fi

statistical and other useful tools through its unique *editorial approach* that was invented by Mustonen in 1979 [10]. It allows the user to create mixtures of computational schemes and natural language documentation within a unified, interactive environment [8, 9, 11].

The newest generation of Survo is Survo R (initially called Muste), which has been implemented as a multiplatform, open source R package [12] that works under Linux, Mac OS X, and Windows. Survo R is freely downloadable from `http://www.survo.fi/`. It brings the editorial approach of Survo inside R. The center of the approach is an *edit field*, where the commands and operations are written and documented. When a user *activates* an operation, it is automatically interpreted and carried out, and the results are written back to the same edit field or external files. Despite the interpretive nature, most of the statistical and other operations of Survo R have been programmed in C language. We will demonstrate the editorial approach by several views of the matrix computations.

The structure of the paper is as follows. First, we present our method of solving Survo puzzles with matrices by going through the necessary phases with two small Survo puzzles. After that, we apply the method in more general form for larger puzzles. We end up with conclusions and a short discussion.

2. Method of solving Survo puzzles with matrices

Our method of solving Survo puzzles with matrices consists of two stages:

(1) *Constructing the code matrix* of the same size as the puzzle under study.
(2) *Analyzing the partitions* of the row sums and the column sums of the puzzle.

The code matrix of Stage 1 will include a unique code for each position of the puzzle. The same code matrix works for all Survo puzzles of the same size, as it depends only on m and n, not on the row sums or the column sums, or their partitions.

The partitions of Stage 2 are analyzed by considering one row and one column of the puzzle at a time.

At both stages, the information is initially coded in a binary form. It is then recoded and transformed to achieve and retain unique coding when the pieces of information are cumulated and combined.

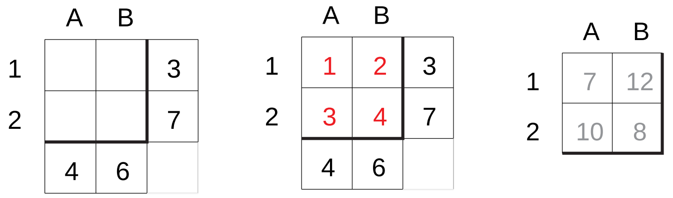
Finally, the results of the two stages are used to obtain the solution of the Survo puzzle, that is, to find the connection between the codes, the positions, and the numbers $1, 2, \ldots, mn$.

2.1 *Stage 1: Constructing the code matrix*

For clarity, we will focus exclusively on open Survo puzzles in the case where $m = n$. In addition, we will begin with a small Survo puzzle, with $m = n = 2$, see Fig. 1 (a). It is an open Survo puzzle, as none of the $mn = 4$ integers (1, 2, 3, 4) are readily given.

The rows of the puzzle are labeled as 1 and 2. Their sums are fixed at 3 and 7, respectively. Similarly, the columns of the puzzle that are labeled as A and B, have sums equal to 4 and 6. The solution of this Survo puzzle appears in Fig. 1 (b). It is most trivial to find, because the only partitions for the sums 3 and 4 are $3 = 1 + 2$ and $4 = 1 + 3$, which implies that the number in the upper left corner (1A) must be 1, and hence the number in the lower right corner (2B) has to be 4. Obviously, there are no other solutions.

However, the point here is the code matrix in Fig. 1 (c), displayed in the same format with the row and the column labels. To make a distinction with the actual numbers, the codes are written in gray color, and the sums are omitted.



(a) Survo puzzle. (b) Solution. (c) Code matrix.

Figure 1. An open 2×2 Survo puzzle, its solution, and the 2×2 code matrix.

The code matrix is constructed from binary matrices

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \ \mathbf{X}_2 = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}, \ \mathbf{X}_A = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \text{ and } \mathbf{X}_B = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix},$$

where the ones indicate the elements of the row or column in question. We will process one row and one column at a time, to take advantage of the fact that they always have one common position in the puzzle. That will give us a unique code independently of the size of the puzzle.

Let us proceed in order and begin from the row 1 and the column A. The first phase is then to work with the matrices $\mathbf{X}_1$ and $\mathbf{X}_A$. We combine them by computing the sum

$$\mathbf{X}_{1A} = \mathbf{X}_1 + 2\mathbf{X}_A = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 2 & 0 \end{bmatrix},$$

which gives us four unique codes, because $\mathbf{X}_A$ is first multiplied by 2. (Otherwise we would get only two unique codes.) Actually, four unique codes would be enough for the 2×2 code matrix, but we continue with another pair of matrices to show the additional phases that lead to the code matrix in Fig. 1 (c). Without these phases the idea of the code matrix would be difficult to generalize for larger puzzles.

Hence we consider the row 2 and the column B similarly and obtain

$$\mathbf{X}_{2B} = \mathbf{X}_2 + 2\mathbf{X}_B = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 2 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 1 & 3 \end{bmatrix},$$

another set of similar codes. The next phase is to combine the two matrices

$$\mathbf{X}_{1A} = \begin{bmatrix} 3 & 1 \\ 2 & 0 \end{bmatrix} \text{ and } \mathbf{X}_{2B} = \begin{bmatrix} 0 & 2 \\ 1 & 3 \end{bmatrix},$$

which will take place by the Hadamard product. Before that, the elements of $\mathbf{X}_{1A}$ and $\mathbf{X}_{2B}$ must be recoded so that the result is also a set of unique codes. (It is easy to see that a Hadamard product without recoding would destroy this requirement.)

The recoding approach may be chosen in various ways, but in this phase a handy alternative is to employ functions $f_1(x) = 2x + 1$ and $f_2(x) = 2^x$, where x refers to the elements of the matrices. Proceeding with the transformations so that $f_1(x)$ is applied for the elements of $\mathbf{X}_{1A}$ and $f_2(x)$ for the elements of $\mathbf{X}_{2B}$ we obtain new matrices (retaining the same names for simplicity)

$$\mathbf{X}_{1A} = \begin{bmatrix} 7 & 3 \\ 5 & 1 \end{bmatrix} \text{ and } \mathbf{X}_{2B} = \begin{bmatrix} 1 & 4 \\ 2 & 8 \end{bmatrix}.$$

The codes in $\mathbf{X}_{1A}$ refer to the following positions in the Survo puzzle:

- 7 refers to the row 1 and column A (position 1A),
- 3 refers to the row 1,
- 5 refers to the column A, and
- 1 refers to the other position, outside of the row 1 and the column A,

while, respectively, in $\mathbf{X}_{2B}$:

- 1 refers to the other position, outside of the row 2 and the column B,
- 4 refers to the column B,
- 2 refers to the row 2, and
- 8 refers to the row 2 and column B (position 2B).

Obviously, it is sufficient to have four codes to refer to four positions, so we should combine the above information in one matrix. For example, the combination of 7 and 1 will not change anything, because 7 already refers to a unique position 1A, whereas the combination of 3 and 4 will now uniquely refer to row 1 *and* column B (i.e., position 1B). Hence, to achieve this, we combine the matrices with the Hadamard product and obtain the final code matrix

$$\mathbf{X}_{1A2B} = \mathbf{X}_{1A} \circ \mathbf{X}_{2B} = \begin{bmatrix} 7 & 12 \\ 10 & 8 \end{bmatrix},$$

which works for any 2×2 Survo puzzle. Its four codes refer exactly to the following four positions of a Survo puzzle:

- 7 refers to the position 1A,
- 12 refers to 1B,
- 10 refers to 2A, and
- 8 refers to 2B.

Later, we will see that these codes appear also in the larger code matrices, but new sets of codes are needed as soon as the size of the puzzle increases.

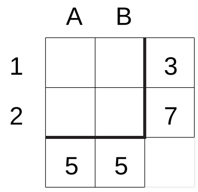
2.2 *Stage 2: Analyzing the partitions*

The previous example is too simple for Stage 2, as it does not contain alternative partitions for any row or column sums. Therefore we will now study another 2×2 Survo puzzle, shown in Fig. 2. This puzzle does not have a unique solution, however, because the columns A and B (with the sums equal to 5) could be interchanged. Both solutions appear in Fig. 2, together with the code matrix created at Stage 1. The puzzle will serve as the minimal example of several partitions, as there are two possible partitions of two distinct parts for the column sum 5, denoted by $P_A = \{1, 4\}$ and $\{2, 3\}$ (or similarly for P_B). The row sums 3 and 7 have only one partition each, namely, $P_1 = \{1, 2\}$ and $P_2 = \{3, 4\}$.

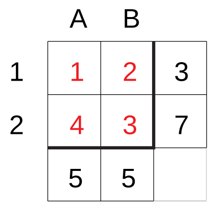
The essential idea is to consider the partitions as $p_i \times mn$ binary matrices, where p_i is the number of the partitions for each of the sums, $i = 1, 2, \ldots, m+n$. Each row of these matrices includes mn binary digits, where the ones indicate the numbers of the partition.

We will analyze two such binary matrices at a time, one for a row and one for a column of the Survo puzzle, and combine them with similar recodings as with the code matrix. The crucial difference is that the dimensions of the matrices will now vary, depending on the number of the partitions. Finally we will obtain a unique set of codes that will be compared with the code matrix to find the solution of the puzzle.

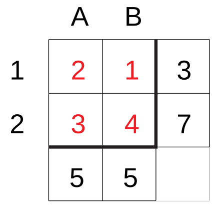
We will again begin from the row 1 and the column A. To stress the distinction with



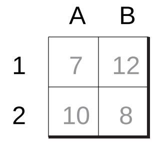
(a) Survo puzzle.



(b) Solution 1.



(c) Solution 2.



(d) Code matrix.

Figure 2. An open 2×2 Survo puzzle, its solutions, and the 2×2 code matrix.

the code matrix, we denote the corresponding binary partition matrices by

$$\mathbf{P}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{P}_A = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

The idea and the implementation of our method is based on computational experiments with the matrix interpreter and other operations of Survo, and hence the way of using Survo deserves to be demonstrated by a few concrete examples[1]. Hence, in Fig. 3, we have a view of the Survo editor, where we create the four binary matrices to be analyzed.

When a Survo command is *activated* (e.g., by a double-click of the mouse), the editor reads and interprets the input from the surrounding *edit field*, then completes the task, and writes the output back to the same field and/or to matrix files or other external files. The leftmost column of the edit field (initially filled by asterisks) is called the *control column*. In Fig. 3, it has no special role, but in some of the further work schemes, it will be used for managing the job flow, making it easier to repeat the steps of the work.

In Fig. 3, we employ the COMB command, which lists various combinatorial entities (here, partitions) in the edit field according to the given specifications (like MAX=4). It will be more seriously needed later in more demanding settings. The MAT commands operate with matrices that are defined in the edit field and saved in matrix files. The rest of the text in Fig. 3 consists of free-form comments and notes written by the user.

```
*COMB P CUR+1    /  P=PARTITIONS,5,2 DISTINCT=1 MIN=1 MAX=4
*Partitions 2 of 5: N[P]=2
*1 4
*2 3
*
*MAT SAVE AS P1  /  only one partition: {1,2}
*1 1 0 0
*
*MAT SAVE AS P2  /  only one partition: {3,4}
*0 0 1 1
*
*MAT SAVE AS PA  /  two partitions: {1,4} and {2,3}
*1 0 0 1
*0 1 1 0
*
*MAT SAVE AS PB  /  two partitions: {2,3} and {1,4}
*0 1 1 0
*1 0 0 1
*
*MAT DIM P1 /* rowP1=1 colP1=4
*MAT DIM PA /* rowPA=2 colPA=4
```

Figure 3. Creating the binary matrices in the edit field of Survo.

[1]The examples in extended form are freely available from `http://www.survo.fi/puzzles/matrix-examples.zip`.

We return to the task of analyzing the partitions. To combine $\mathbf{P}_1$ and $\mathbf{P}_A$ we must make them compatible. This is achieved with the help of three suitably chosen vectors of ones that are further used to create two matrices of ones. The aim is to create a combined matrix $\mathbf{K}$, which has two rows. It is achieved by multiplying the binary partition matrices and the matrices of ones across using the Kronecker product and then summing them together so that $\mathbf{P}_A$ is multiplied by 2 (similarly as with the code matrix at Stage 1):

$$\mathbf{K} = \mathbf{K}_1 + \mathbf{K}_A = (\mathbf{P}_1 \otimes \mathbf{1}\mathbf{1}') + (\mathbf{1}\mathbf{1}' \otimes 2\mathbf{P}_A).$$

These phases are accomplished in Survo as shown in Fig. 4 and Fig. 5.

```
*MAT One1=CON(rowP1,1)    /  dimensions corresponding to P1 and PA
*MAT OneA=CON(rowPA,1)    /  (symbolic dimensions are achieved
*MAT Ones=CON(colP1,1)    /   with the MAT DIM commands above)
*MAT NAME One1 AS 1
*MAT NAME OneA AS 1       /  for simplicity, change the internal
*MAT NAME Ones AS 1       /   names of the matrices to plain "1"
*
*MAT Ones1=One1*Ones'
*MAT OnesA=OneA*Ones'
*
*MAT LOAD Ones1 111 CUR+1 /  matrices have row and column labels...
*MATRIX Ones1
*1*1'
*///        1   2   3   4
*  1        1   1   1   1
*
*MAT LOAD OnesA 111 CUR+1 /  ...that are automatically updated (see below)
*MATRIX OnesA
*1*1'
*///        1   2   3   4
*  1        1   1   1   1
*  2        1   1   1   1
```

Figure 4. Creating suitable vectors and matrices of ones.

```
*MAT K1=KRONECKER(P1,OnesA)
*MAT KA=KRONECKER(Ones1,2*PA)
*
*MAT LOAD K1 111 CUR+1
*MATRIX K1
*KRONECKER(P1,1*1')
*///      1*1 1*2 1*3 1*4 2*1 2*2 2*3 2*4 3*1 3*2 3*3 3*4 4*1 4*2 4*3 4*4
*1*1        1   1   1   1   1   1   1   1   0   0   0   0   0   0   0   0
*1*2        1   1   1   1   1   1   1   1   0   0   0   0   0   0   0   0
*
*MAT LOAD KA 111 CUR+1
*MATRIX KA
*KRONECKER(1*1',2*PA)
*///      1*1 1*2 1*3 1*4 2*1 2*2 2*3 2*4 3*1 3*2 3*3 3*4 4*1 4*2 4*3 4*4
*1*1        2   0   0   2   2   0   0   2   2   0   0   2   2   0   0   2
*1*2        0   2   2   0   0   2   2   0   0   2   2   0   0   2   2   0
*
*MAT K=K1+KA  /  Now the dimensions match, so we can compute the sum.
*MAT LOAD K 111 CUR+1
*MATRIX K
*KRONECKER(P1,1*1')+KRONECKER(1*1',2*PA)
*///      1*1 1*2 1*3 1*4 2*1 2*2 2*3 2*4 3*1 3*2 3*3 3*4 4*1 4*2 4*3 4*4
*1*1        3   1   1   3   3   1   1   3   2   0   0   2   2   0   0   2
*1*2        1   3   3   1   1   3   3   1   0   2   2   0   0   2   2   0
```

Figure 5. Combining the matrices with the Kronecker product.

The resulting matrix $\mathbf{K}$ (see Fig. 5) looks promising, but it contains too many numbers. The reason is that the Kronecker product also operates columnwise, which is not needed in our method. We are only interested in the *principal columns* of the matrix $\mathbf{K}$, automatically labeled as `1*1`, `2*2`, `3*3` and `4*4` by the matrix interpreter of Survo in Fig. 5.

We extract the principal columns of $\mathbf{K}$ by forming another matrix

$$\mathbf{J} = (\mathbf{I} \otimes \mathbf{1}) \circ (\mathbf{1} \otimes \mathbf{I}),$$

where $\mathbf{I}$ refers to a 4×4 identity matrix and $\mathbf{1}$ is a 4×1 vector of ones. Multiplying $\mathbf{K}$ from the right by $\mathbf{J}$ gives us the combined 2×4 matrix (see Fig. 6)

$$\mathbf{P}_{1A} = \mathbf{KJ} = \big[(\mathbf{P}_1 \otimes \mathbf{11}') + (\mathbf{11}' \otimes 2\mathbf{P}_A)\big]\big[(\mathbf{I} \otimes \mathbf{1}) \circ (\mathbf{1} \otimes \mathbf{I})\big].$$

```
*MAT I=IDN(colP1,colP1)           /  identity matrix (4 x 4)
*MAT One=CON(colP1,1)             /  and unit vector (4 x 1)
*
*MAT J1=KRONECKER(I,One)          /  Kronecker product 1,
*MAT J2=KRONECKER(One,I)          /  Kronecker product 2 and
*MAT J=#HADAMARD(J1,J2)           /  their Hadamard product
*MAT LOAD J 111 CUR+1             /  (16 x 4)
*MATRIX J
*Hadamard(KRONECKER(IDN,CON),KRONECKER(CON,IDN))
*///      1*1 2*1 3*1 4*1
*1*1        1   0   0   0
*1*2        0   0   0   0
 [...] (some rows omitted to save space)
*2*2        0   1   0   0
 [...] (some rows omitted to save space)
*4*3        0   0   0   0
*4*4        0   0   0   1
*
*MAT P1A=K*J
*MAT LOAD P1A 111 CUR+1
*MATRIX P1A
*///      1*1 2*1 3*1 4*1
*1*1        3   1   0   2
*1*2        1   3   2   0
```

Figure 6. Extracting the principal columns of the Kronecker product.

The matrix interpreter of Survo includes a SUB function to extract the principal columns in a more straight-forward way by using the matrix $\mathbf{H} = \mathrm{vec}(\mathbf{I})'$, which is shown briefly in Fig. 7.

```
*MAT H=VEC(I)'          /  vectorize and transpose the identity matrix
*MAT LOAD H 111 CUR+1
*MATRIX H
*VEC(IDN)'
*///        1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
*  1        1   0   0   0   0   1   0   0   0   0   1   0   0   0   0   1
*
*MAT P1A=SUB(K,*,H)     /  extract those columns of K indicated by H (* = all rows)
*MAT LOAD P1A 111 CUR+1
*MATRIX P1A
*SUB(KRONECKER(P1,1*1')+KRONECKER(1*1',2*PA),*,H)
*///      1*1 2*2 3*3 4*4
*1*1        3   1   0   2
*1*2        1   3   2   0
```

Figure 7. Another way of extracting the principal columns.

Now, we have completed the analysis of the binary partition matrices $\mathbf{P}_1$ and $\mathbf{P}_A$. The analysis of the next pair ($\mathbf{P}_2$ and $\mathbf{P}_B$) would follow similarly and produce the combined matrix

$$\mathbf{P}_{2B} = \big[(\mathbf{P}_2 \otimes \mathbf{11}') + (\mathbf{11}' \otimes 2\mathbf{P}_B)\big]\big[(\mathbf{I} \otimes \mathbf{1}) \circ (\mathbf{1} \otimes \mathbf{I})\big],$$

where the dimensions of the unit vectors and the identity matrix now depend on the dimensions of $\mathbf{P}_2$ and $\mathbf{P}_B$. The details are omitted to save space. Instead, in Fig. 8, we continue from the further recoding phase, where we transform the matrices $\mathbf{P}_{1A}$ and $\mathbf{P}_{2B}$ using the simple functions introduced at Stage 1. For simplicity, we will again retain the names of the matrices.

```
*MAT TRANSFORM P1A BY 2*X#+1  /  further recoding: [1,3,5,7]
*MAT TRANSFORM P2B BY 2^X#    /  further recoding: [1,2,4,8]
*
*MAT LOAD P1A 111 CUR+1
*MATRIX P1A
*T(P1A_by_2*X#+1)
*///      1*1 2*2 3*3 4*4
*1*1        7   3   1   5
*1*2        3   7   5   1
*
*MAT LOAD P2B 111 CUR+1
*MATRIX P2B
*T(P2B_by_2^X#)
*///      1*1 2*2 3*3 4*4
*1*1        1   4   8   2
*1*2        4   1   2   8
```

Figure 8. Further recoding of the two matrices.

In this case, the matrices $\mathbf{P}_{1A}$ and $\mathbf{P}_{2B}$ have equal dimensions, and it might be temptating to use the Hadamard product to combine them. However, in the more general situations the dimensions will be unequal (because of the varying number of partitions), and in any case, we must consider all the combinations of the rows. Therefore, in Fig. 9, we apply the Kronecker product to form a new matrix $\mathbf{P}_{1A2B}$, again extracting its principal columns only.

```
*MAT P1A2B=SUB(KRONECKER(P1A,P2B),*,H)    /  combine and extract the columns
*MAT CLABELS NUM(1) TO P1A2B              /  put column labels "1,2,3,4"
*MAT RLABELS r      TO P1A2B              /  and row labels "r1,r2,r3,r4"
*MAT LOAD P1A2B 111 CUR+1
*MATRIX P1A2B
*SUB(KRONECKER(T(P1A_by_2*X#+1),T(P2B_by_2^X#)),*,H)
*///        1   2   3   4
*r1         7  12   8  10
*r2        28   3   2  40   <- illogical combination
*r3         3  28  40   2   <- illogical combination
*r4        12   7  10   8
```

Figure 9. Combining the recoded matrices and displaying the result.

As we can see from Fig. 9, the combined partition matrix $\mathbf{P}_{1A2B}$ includes two illogical combinations (on rows labeled `r2` and `r3`). For example, recalling the explanations of the codes from Stage 1, the code 28 (product of codes 7 and 4), would refer to the position 1A *and* column B, which would be clearly impossible. With larger puzzles, there will unavoidably be a huge number of illogical combinations, which have to be discarded while analyzing the partitions. It will be straight-forward, because the logical codes are always provided with the code matrix.

Using the matrix notation, the operations in Fig. 9 can be written in the form

$$\mathbf{P}_{1A2B} = (\mathbf{P}_{1A} \otimes \mathbf{P}_{2B})\mathbf{J},$$

where $\mathbf{J} = (\mathbf{I} \otimes \mathbf{1}) \circ (\mathbf{1} \otimes \mathbf{I})$, but the shortest way to express this is the Khatri–Rao product [7], which we denote by

$$\mathbf{P}_{1A2B} = \mathbf{P}_{1A} \odot \mathbf{P}_{2B}.$$

The Khatri–Rao product seems to appear in the matrix literature in various forms and notations [13]. The form we apply here could be called a "row-wise Kronecker product", as we are not interested in the products of the columns. It is also referred to as the "Khatri–Rao of first kind product" by [13, p. 765].

2.3 *Finding the solution*

The solution of the Survo puzzle in Fig. 2 (a) can now be found using the code matrix $\mathbf{X}_{1A2B}$ created at Stage 1 (see Fig. 2 (d)) and the matrix $\mathbf{P}_{1A2B}$ created at Stage 2 (see Fig. 9). For an easier reference, we have collected the matrices (without the two illogical combinations) in Fig. 10 together with the two solutions (without the row and the column sums).

Now, it is important to look at the labels of the matrices in Fig. 10, especially of the matrix $\mathbf{P}_{1A2B}$, because its column labels (1, 2, 3, 4) represent the numbers that should be put in the four positions of the Survo puzzle. The numbers are in the natural order, which follows from the way how the partition matrices were originally created. Therefore we can find the both solutions easily just by looking at the matrices. Let us begin from the codes on the row `r1` and put

- number 1 to the position indicated by the code 7 in the code matrix, that is, 1A,
- number 2 to 1B (code 12),
- number 3 to 2B (code 8), and
- number 4 to 2A (code 10).

This is clearly the Solution 1. Similarly, the codes on the row `r4` give the Solution 2.

```
Code matrix:      Combined partitions:         Solution 1:     Solution 2:
      A   B             1   2   3   4                A   B           A   B
1     7  12       r1    7  12   8  10          1     1   2     1     2   1
2    10   8       r4   12   7  10   8          2     4   3     2     3   4
```

Figure 10. The code matrix, the combined partition matrix, and the two solutions of a 2×2 Survo puzzle.

2.4 *Constructing code matrices for larger puzzles*

With the 2×2 puzzles we saw that all the four codes were unique after the very first phase of combining the binary information. When creating code matrices for larger puzzles, most of the codes will not be unique after the first phase. Importantly, however, the same coding system works for all puzzles.

To illustrate this, we show the phases of the code matrix for 3×3 Survo puzzles in Fig. 11. In the first two phases (a) and (b), code 3 is the only unique code, while codes 1 and 2 both appear twice, and code 0 appears four times. The next two phases (c) and (d) show the transformed versions of (a) and (b), respectively.

The last phase in Fig. 11 (e) is the 3×3 code matrix, obtained from the previous two phases (c) and (d) by the Hadamard product. In its upper-left corner we can see the unique codes 7, 12, 10, and 8, that were the only codes in the 2×2 case. In addition, the rest of the codes (1, 2, 3, 4, and 5) have become unique as well, thus identifying the exact positions 3C, 2C, 1C, 3B, and 3A, respectively, in 3×3 puzzles.

We can conclude that solving a 3×3 puzzle requires combining two pairs of recoded binary matrices, whereas one pair would have been enough with the 2×2 puzzle. Similarly, three pairs will be required with a 4×4 puzzle.

	A	B	C
1	3	1	1
2	2	0	0
3	2	0	0

(a) Phase 1A.

	A	B	C
1	0	2	0
2	1	3	1
3	0	2	0

(b) Phase 2B.

	A	B	C
1	7	3	3
2	5	1	1
3	5	1	1

(c) Recoded 1A.

	A	B	C
1	1	4	1
2	2	8	2
3	1	4	1

(d) Recoded 2B.

	A	B	C
1	7	12	3
2	10	8	2
3	5	4	1

(e) Code matrix.

Figure 11. The phases of the code matrix for 3×3 Survo puzzles.

To prepare for the 4×4 case, we construct the code matrix for 4×4 Survo puzzles using the matrix interpreter of Survo (see Fig. 12). To expand the coding scheme, we have chosen a new set of codes (1, 9, 11, 13) so that the combinations of the codes will again remain unique after the Hadamard products (or Kronecker and Khatri–Rao products, when analyzing the partitions).

```
*/ACTIVATE +  /  This activates all rows that have a '+' in the control column.
+MAT X=ZER(4,4)  /  Create a 4 x 4 null matrix X with labels
+MAT X(0,1)="A"  /  corresponding to the Survo puzzle.
+MAT X(0,2)="B"
+MAT X(0,3)="C"
+MAT X(0,4)="D"
+MAT RLABELS NUM(1) TO X
+MAT X1=X  / X is the basis for each binary matrix.
+MAT XA=X  /                                        This is how the matrices
+MAT X2=X  / We need three pairs of rows and columns.   X1...XC look like before
+MAT XB=X  / Choose X1 and XA, X2 and XB, X3 and XC.    recoding and combining:
+MAT X3=X  /
+MAT XC=X  /                                        X1:      X2:      X3:
*                                                   1 1 1 1  0 0 0 0  0 0 0 0
*Mark the rows and the columns suitably with ones:  0 0 0 0  1 1 1 1  0 0 0 0
*(I# and J# refer to row and column indices)        0 0 0 0  0 0 0 0  1 1 1 1
*                                                   0 0 0 0  0 0 0 0  0 0 0 0
+MAT #TRANSFORM X1 BY F1 / F1=if(I#=1)then(1)else(0)
+MAT #TRANSFORM XA BY FA / FA=if(J#=1)then(1)else(0)    XA:      XB:      XC:
+MAT #TRANSFORM X2 BY F2 / F2=if(I#=2)then(1)else(0)    1 0 0 0  0 1 0 0  0 0 1 0
+MAT #TRANSFORM XB BY FB / FB=if(J#=2)then(1)else(0)    1 0 0 0  0 1 0 0  0 0 1 0
+MAT #TRANSFORM X3 BY F3 / F3=if(I#=3)then(1)else(0)    1 0 0 0  0 1 0 0  0 0 1 0
+MAT #TRANSFORM XC BY FC / FC=if(J#=3)then(1)else(0)    1 0 0 0  0 1 0 0  0 0 1 0
*
*Recode and combine (see the results on the right):     X1A:     X2B:     X3C:
+MAT X1A=X1+2*XA         / W=if(X#=0)then(01)else(a)    3 1 1 1  0 2 0 0  0 0 2 0
+MAT X2B=X2+2*XB         / a=if(X#=1)then(09)else(b)    2 0 0 0  1 3 1 1  0 0 2 0
+MAT X3C=X3+2*XC         / b=if(X#=2)then(11)else(c)    2 0 0 0  0 2 0 0  1 1 3 1
*U=2*X#+1 V=2^X#         / c=if(X#=3)then(13)else(X#)   2 0 0 0  0 2 0 0  0 0 2 0
*
*Further recoding (the rules U,V,W defined above):
+MAT #TRANSFORM X1A BY U / [1,3,5,7]                    The matrices X1A, X1B, X1C
+MAT #TRANSFORM X2B BY V / [1,2,4,8]                    after the both recodings:
+MAT #TRANSFORM X3C BY W / [1,9,11,13] (new codes!)
*                                                  /  X1A:     X2B:     X3C:
*Combine with Hadamard product X1A o X2B o X3C:    /  7 3 3 3  1 4 1 1  1 1 11 1
+MAT        X1A2B=#HADAMARD(X1A,X2B)               /  5 1 1 1  2 8 2 2  1 1 11 1
+MAT      X1A2B3C=#HADAMARD(X1A2B,X3C)             /  5 1 1 1  1 4 1 1  9 9 13 9
+MAT NAME X1A2B3C AS 4x4                           /  5 1 1 1  1 4 1 1  1 1 11 1
+LOADM X1A2B3C 111 CUR+1  /  The final code matrix
*4x4
*           A   B   C   D
*  1        7  12  33   3
*  2       10   8  22   2
*  3       45  36  13   9
*  4        5   4  11   1
```

Figure 12. Creating the code matrix for 4×4 Survo puzzles.

	A	B	C	
1				15
2				15
3				15
	15	15	15	

(a) Survo puzzle.

	A	B	C	
1	4	9	2	15
2	3	5	7	15
3	8	1	6	15
	15	15	15	

(b) Solution 1.

	A	B	C	
1	1	5	9	15
2	6	7	2	15
3	8	3	4	15
	15	15	15	

(c) Solution 2.

Figure 13. An open 3×3 Survo puzzle and two of its 72 solutions.

3. Solving larger Survo puzzles with matrices

It is obvious that the 2×2 puzzles are very small, but it is not as clear what is meant by a "large Survo puzzle". In principle there is no upper limit for the dimensions m and n, but in practice they are always quite small integers. One of the most typical cases is $m = n = 4$. An open 5×5 Survo puzzle is quite a challenge, as it is not even known, how many uniquely solvable 5×5 Survo puzzles do exist [2].

In the following, we use our method to solve two larger Survo puzzles. The first one is a special, open 3×3 Survo puzzle, as it can also be considered as a Magic square. The second one is a typical, open 4×4 Survo puzzle.

3.1 *Solving a 3×3 Magic Survo puzzle*

Magic squares can be seen as special cases of Survo puzzles, when $m = n$ and all the row and column sums are equal to the same number. Here, we consider briefly a 3×3 puzzle with these special properties (see Fig. 13). Using our method we will find 72 alternative solutions to the puzzle. Two examples of solutions are shown in Fig. 13 (b) and (c).

Only eight of the 72 solutions have an additional property required for Magic squares, that is, the numbers in their main diagonals also add up to the same number, called the Magic sum $n(n^2 + 1)/2 = 15$.

```
          1  2  3  4  5  6  7  8  9
1 5 9  :  1  0  0  0  1  0  0  0  1
1 6 8  :  1  0  0  0  0  1  0  1  0
2 4 9  :  0  1  0  1  0  0  0  0  1
2 5 8  :  0  1  0  0  1  0  0  1  0
2 6 7  :  0  1  0  0  0  1  1  0  0
3 4 8  :  0  0  1  1  0  0  0  1  0
3 5 7  :  0  0  1  0  1  0  1  0  0
4 5 6  :  0  0  0  1  1  1  0  0  0
```

Figure 14. The partitions of 15 with their binary forms in the edit field of Survo.

In Fig. 14, we have listed the partitions of 15 consisting of three distinct parts together with their binary forms. There are a total of eight such partitions, exactly one for each of the rows, columns, and main diagonals needed for the Magic square.

Proceeding similarly as with the smaller example, we obtain the binary partition matrices $\mathbf{P}_1$, $\mathbf{P}_A$, $\mathbf{P}_2$, and $\mathbf{P}_B$, which are, of course, identical in this special case. Combining them pairwise leads to the (identical) matrices $\mathbf{P}_{1A}$ and $\mathbf{P}_{2B}$ of $8^2 = 64$ rows. Next, we use the simple functions introduced at Stage 1 to transform the elements of the matrices (again retaining the names of the matrices for simplicity). Combining the matrices with the Khatri–Rao product

$$\mathbf{P}_{1A2B} = \mathbf{P}_{1A} \odot \mathbf{P}_{2B}$$

produces a matrix of $64^2 = 4096$ rows. Most of the rows include illogical codes, similarly as we saw with the smaller example. To discard the illogical codes we must compare the rows of $\mathbf{P}_{1A2B}$ with the 3×3 code matrix in Fig. 11 (e). (This phase will be explained in more detail with the 4×4 example.) The resulting matrix has exactly 72 rows, representing the possible solutions in coded form. Those eight that also fulfill the diagonal requirement of the Magic square, are listed in Fig. 15. They all have number 5 in the position indicated by the code 8, which refers to the middle position 2B, see Fig. 11 (e). Comparing these eight solutions with the code matrix in Fig. 11 (e) reveals that the solution displayed in Fig. 13 (b) corresponds to the row labeled `r1393`.

	1	2	3	4	5	6	7	8	9
r817	12	5	2	1	8	7	10	3	4
r881	12	1	10	5	8	3	2	7	4
r1329	4	7	2	3	8	5	10	1	12
r1393	4	3	10	7	8	1	2	5	12
r2119	10	3	4	1	8	7	12	5	2
r2183	2	7	4	5	8	3	12	1	10
r2631	10	1	12	3	8	5	4	7	2
r2695	2	5	12	7	8	1	4	3	10

Figure 15. The codes of the combined partitions that correspond to the 3×3 Magic square.

3.2 *Solving a 4×4 Survo puzzle*

Finally, we use our method to solve a typical, open 4×4 Survo puzzle [14] that is not an extremely difficult puzzle, but definitely far from trivial. In Fig. 16, we have the puzzle, the solution, and the code matrix.

We will follow the same logic as with the small puzzles, but in order to operate with a substantially larger number of partitions, we need to employ various other data analytic operations of Survo as well. These include the COMB command, which was briefly shown earlier, accompanied with several FILE operations for creating and managing Survo data files as well as VAR, TRANSFORM, and VARSTAT commands for making transformations and summaries of variables.

In Fig. 17, we create the binary partition matrices $\mathbf{P}_1$ and $\mathbf{P}_A$ for the row 1 and column A of the puzzle. The partitions (61 for 1 and 23 for A) are saved in a text file by the COMB command and then moved in data files, transformed into a binary format and saved in

	A	B	C	D	
1					27
2					12
3					52
4					45
	20	45	31	40	

(a) Survo puzzle.

	A	B	C	D	
1	3	10	6	8	27
2	1	5	2	4	12
3	9	16	12	15	52
4	7	14	11	13	45
	20	45	31	40	

(b) Solution.

	A	B	C	D
1	7	12	33	3
2	10	8	22	2
3	45	36	13	9
4	5	4	11	1

(c) Code matrix.

Figure 16. An open 4×4 Survo puzzle, its solution, and the corresponding code matrix.

```
*/ACTIVATE +  /  This activates all rows that have a '+' in the control column.
*---- Row 1: ----------
+COMB P1 TO P1.TXT / P1=PARTITIONS,27,4  MIN=1 MAX=16 DISTINCT=1
+FILE MAKE D1,16,0,X,1  / Create a data file D1 with variables X1,X2,...,X16
+FILE SAVE P1.TXT TO D1 / Save partitions from the text file to the data
+XALL D1,X1,16          / Move the numbers into their right positions
+TRANSFORM D1 BY BINARY / Transform: BINARY=if(X=MISSING)then(0)else(1)
+MAT SAVE DATA D1 TO P1 / Save the data in a binary matrix P1
*---- Column A: -------
+COMB PA TO PA.TXT / PA=PARTITIONS,20,4
+FILE MAKE DA,16,0,X,1
+FILE SAVE PA.TXT TO DA
+XALL DA,X1,16
+TRANSFORM DA BY BINARY
+MAT SAVE DATA DA TO PA
*---- Matrices of ones:
+MAT DIM P1 /* rowP1=61 colP1=16
+MAT DIM PA /* rowPA=23 colPA=16
+MAT One1!=CON(rowP1,1)
+MAT OneA!=CON(rowPA,1)
+MAT Ones!=CON(colP1,1)
+MAT Ones1=One1*Ones'
+MAT OnesA=OneA*Ones'
 [...] (some rows omitted to save space)
+MAT H=VEC(IDN(16,16))' / for extracting the principal columns
+MAT P1A=SUB(KRONECKER(P1,OnesA)+KRONECKER(Ones1,2*PA),*,H)
+MAT P2B=SUB(KRONECKER(P2,OnesB)+KRONECKER(Ones2,2*PB),*,H)
+MAT P3C=SUB(KRONECKER(P3,OnesC)+KRONECKER(Ones3,2*PC),*,H)
```

Figure 17. Creating and combining the matrices of partitions.

matrix form for subsequent operations. The two other required pairs (not shown) are handled similarly, leading to the matrices $\mathbf{P}_2$, $\mathbf{P}_B$, $\mathbf{P}_3$, and $\mathbf{P}_C$, with 2, 38, 9, and 79 partitions, respectively. In the end of Fig. 17, they are combined to form the matrices $\mathbf{P}_{1A}$, $\mathbf{P}_{2B}$, and $\mathbf{P}_{3C}$, using the Kronecker product and extracting the principal columns as described with the simpler examples earlier.

The number of the partitions may seem reasonable, or even small, but the total number of their products will soon be large. In the above matrices, we have $61 \cdot 23 = 1403$, $2 \cdot 38 = 76$ and $9 \cdot 79 = 711$ rows. Hence, their Khatri–Rao product would have $1403 \cdot 76 \cdot 711$, that is, over 75 million rows. Only one row includes the right codes (exactly those included in the code matrix), assuming that the puzzle has a unique solution.

To avoid too large dimensions, we remove the illogical combinations after each Kronecker or Khatri–Rao product. In each phase of Stage 2, we apply a set of conditions based on the codes of the corresponding phase of Stage 1, check through the combinations and remove the cases that do not meet those conditions. As we have seen, the set of such conditions (the codes in the code matrix) will be updated each time we combine information about a new pair of a row and a column. In the 4×4 case, the three phases of the code matrix are

$$\mathbf{X}_{1A} = \begin{bmatrix} 3 & 1 & 1 & 1 \\ 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{X}_{1A2B} = \begin{bmatrix} 7 & 12 & 3 & 3 \\ 10 & 8 & 2 & 2 \\ 5 & 4 & 1 & 1 \\ 5 & 4 & 1 & 1 \end{bmatrix}, \text{ and } \mathbf{X}_{1A2B3C} = \begin{bmatrix} 7 & 12 & 33 & 3 \\ 10 & 8 & 22 & 2 \\ 45 & 36 & 13 & 9 \\ 5 & 4 & 11 & 1 \end{bmatrix}.$$

The final set of the conditions (the code matrix $\mathbf{X}_{1A2B3C}$) consists of 16 unique codes. Only one of the remaining rows of the combined partition matrix will meet all those conditions and lead to the solution of the puzzle.

The illogical combinations are removed in Survo by moving the matrices to data files, applying the conditions through certain data analytic operations, and then saving the selected rows (the logical combinations) back in the matrix form. These phases are quite similar with each other (only the number of the conditions varies), and hence, to save space, the details are omitted from here except the final one (see Fig. 18), which reveals the codes for the solution.

Let us briefly describe the flow of the intermediate phases. After the Kronecker products (see the end of Fig. 17), the illogical combinations are removed, causing the number of the rows in the matrices $\mathbf{P}_{1A}$, $\mathbf{P}_{2B}$, and $\mathbf{P}_{3C}$ to be reduced from 1403 to 695, from 76 to 25, and from 711 to 405, respectively. This is achieved by applying the conditions based on $\mathbf{X}_{1A}$: on each row, the only unique code 3 must appear once, and the codes 1 and 2 must appear three times each, otherwise the row will be discarded.

Hence, instead of 75 million rows, there would be about 7 million $(695 \cdot 25 \cdot 405)$ rows, if the three matrices were combined by Khatri–Rao products. However, the reduction of the illogical combinations will be maximized by proceeding with one matrix product at a time.

In the next phase, we use the simple functions introduced at Stage 1 to transform the elements of the matrices $\mathbf{P}_{1A}$, $\mathbf{P}_{2B}$, and $\mathbf{P}_{3C}$ (once again retaining the names of the matrices for simplicity). Combining then the first two with the Khatri–Rao product

$$\mathbf{P}_{1A2B} = \mathbf{P}_{1A} \odot \mathbf{P}_{2B}$$

produces a matrix of $695 \cdot 25 = 17375$ rows, which are further reduced, now applying the conditions based on $\mathbf{X}_{1A2B}$: on each row, the four unique codes 7, 8, 10, and 12 must appear only once, and the codes 2, 3, 4, and 5 must appear twice, otherwise the row will be discarded. Here, almost all the rows of $\mathbf{P}_{1A2B}$ represent illogical combinations, as only *three* of them (of the 17375) fulfill the conditions and remain for further processing.

The final phase is displayed in detail in Fig. 18. It begins by the computation of the last Khatri–Rao product needed, that is,

$$\mathbf{P}_{1A2B3C} = \mathbf{P}_{1A2B} \odot \mathbf{P}_{3C},$$

which has $3 \cdot 405 = 1215$ rows. In this phase, the conditions will be most stringent, as all 16 codes of the code matrix $\mathbf{X}_{1A2B3C}$ must appear exactly once. After the conditions have been applied, the resulting reduced version of the matrix $\mathbf{P}_{1A2B3C}$ does have only one row, as the last line in the Fig. 18 indicates.

```
+MAT P1A2B3C=#RAO_KHATRI(P1A2B,P3C)
+MAT DIM P1A2B3C /* rowP1A2B3C=1215 colP1A2B3C=16
*
+FILE SAVE MAT P1A2B3C TO P1A2B3C /  save the matrix to a data file
+FILE MASK P1A2B3C,CASE,1,-       /  mask the case ID out of computations
*Compute the number of each of the codes for all observations of the data:
+VARSTAT P1A2B3C / VARSTAT=N1:1,N2:1,N3:1,N4:1,N5:1,N7:1,N8:1,N10:1,N12:1,&
*                         N9:1,N11:1,N13:1,N22:1,N33:1,N36:1,N45:1
*  N1=#VAL,1    N2=#VAL,2    N3=#VAL,3    N4=#VAL,4    N5=#VAL,5
*  N7=#VAL,7    N8=#VAL,8    N10=#VAL,10  N12=#VAL,12
*  N9=#VAL,9    N11=#VAL,11  N13=#VAL,13  N22=#VAL,22
*  N33=#VAL,33  N36=#VAL,36  N45=#VAL,45
*.........................
+FILE MASK P1A2B3C,CASE,1,A       /  put the case ID back
+FILE MASK P1A2B3C,N1,1,-...      /  mask the numbering variables out
*Save the reduced data back in to a matrix - applying the 16 conditions:
+MAT SAVE DATA P1A2B3C TO P1A2B3C / SELECT=A*B*C*D*E*F*G*H*I*J*K*L*M*N*O*P
*  A=N1,1  B=N2,1 C=N3,1  D=N4,1  E=N5,1  F=N7,1  G=N8,1  H=N10,1
*  I=N12,1 J=N9,1 K=N11,1 L=N13,1 M=N22,1 N=N33,1 O=N36,1 P=N45,1
+MAT DIM P1A2B3C /* rowP1A2B3C=1 colP1A2B3C=16
```

Figure 18. Applying the conditions at the final phase of Stage 2.

Removing the illogical combinations after each Kronecker or Khatri–Rao product is worthwhile, since instead of 75 million or 7 million rows, we have analyzed only $17375 + 1215 = 18590$ rows of combined partitions.

Similarly with the smaller example, we can find the solution easily by looking at the matrices (see Fig. 19). Clearly, we may put

- number 1 to the position indicated by the code 10 in the code matrix, that is, 2A,
- number 2 to 2C (code 22),
- number 3 to 1A (code 7),
- ... (numbers $4 - 14$ similarly),
- number 15 to 3D (code 9), and
- number 16 to 3B (code 36),

and we have solved this 4×4 Survo puzzle.

```
        The only combined partition that fulfills all the 16 conditions:
number:  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
  code: 10  22   7   2   8  33   5   3  45  12  11  13   1   4   9  36

        Code matrix:                          Solution:
             A   B   C   D                         A   B   C   D
         1   7  12  33   3                     1   3  10   6   8
         2  10   8  22   2                     2   1   5   2   4
         3  45  36  13   9                     3   9  16  12  15
         4   5   4  11   1                     4   7  14  11  13
```

Figure 19. The combined partition, the code matrix, and the solution of a 4×4 Survo puzzle.

4. Conclusions and discussion

Solving a Survo puzzle can be clearly formulated as a combinatorial matrix problem, which involves restricted integer partitions with distinct parts. For this problem, we have presented a computational method that is based on working with matrices and binary coded partitions. Although we have focused exclusively on open Survo puzzles in the case where $m = n$, it is possible to generalize our method to cover also other cases, i.e., puzzles where $m \neq n$ or puzzles where some numbers are readily given.

In general, we note that our method is not a computationally efficient way of solving Survo puzzles. Various fast algorithms have been suggested, perhaps the best one still being the original one, developed and programmed in C language by Mustonen [2].

However, our method, which is based on experimenting with the matrix interpreter and other data analytic tools of Survo and transforming the binary partition information in various ways, appears interesting because of its connections to matrices. Through a few concrete examples we have shown how our method is strongly related to the Hadamard, Kronecker, and especially the Khatri–Rao product [7, 13]. In addition, our method is useful for teaching and learning purposes, as each phase and its intermediate results can be easily displayed and studied in detail.

The editorial approach [10] has been the core of Survo for several versions and generations, including the current Survo R [12] that we have used in our computations. The approach binds together the matrix analyses and other operations needed in a particular application. At the same time, it provides an efficient and free-form way of documenting the steps of the research as well as repeating them easily, or even automatically.

Acknowledgements

The authors are grateful for Seppo Mustonen and Simo Puntanen for their comments. The suggestions by an anonymous reviewer were useful in revising the manuscript. The work of Dr Sund was supported by a research grant from the Finnish Cultural Foundation.

References

[1] Mustonen S. On certain cross sum puzzles. http://www.survo.fi/papers/puzzles.pdf. 2006.

[2] Mustonen S. Enumeration of uniquely solvable open Survo puzzles. http://www.survo.fi/papers/enum_survo_puzzles.pdf. 2007.

[3] Andrews GE, Eriksson K. Integer Partitions. Cambridge, UK: Cambridge University Press; 2004.

[4] Lehmer DH. Two nonexistence theorems on partitions. Bulletin of the American Mathematical Society. 1946;52(6):538–544.

[5] Andrews WS. Magic Squares and Cubes. Chicago: Open Court Publishing Company; 1908.

[6] Gustafson K, Styan GPH. Superstochastic matrices and magic Markov chains. Linear Algebra and its Applications. 2009;430:2705–2715.

[7] Khatri CG, Rao CR. Solutions to some functional equations and their applications to characterization of probability distributions. Sankhyā, Ser A. 1968;30:167–180.

[8] Mustonen S. Survo – An Integrated Environment for Statistical Computing and Related Areas. Helsinki, Finland: Survo Systems; 1992; http://www.survo.fi/books/1992/Survo_Book_1992_with_comments.pdf.

[9] Mustonen S. Matrix computations in Survo. Department of Mathematical Sciences, University of Tampere, Finland; 1999. Proceedings of the 8th International Workshop on Matrices and Statistics (IWMS 1999); http://www.helsinki.fi/survo/matrix99.html.

[10] Mustonen S. On interactive statistical data processing. Scandinavian Journal of Statistics. 1981;8:129–136.

[11] Vehkalahti K. Leaving useful traces when working with matrices. Research Letters in the Information and Mathematical Sciences. 2005;8:143–154; Proceedings of the 14th International Workshop on Matrices and Statistics (IWMS 2005).

[12] Sund R, Vehkalahti K, Mustonen S. Muste – editorial computing environment within R; 2012. Proceedings of COMPSTAT 2012, 20th International Conference on Computational Statistics, 27–31 August 2012, Limassol, Cyprus, http://www.survo.fi/muste/publications/sundetal2012.pdf.

[13] Al Zhour Z, Kiliçman A. Some new connections between matrix products for partitioned and non-partitioned matrices. Computers and Mathematics with Applications. 2007;54:763–784.

[14] Mustonen S. Problems of February 8, 2010. http://www.survo.fi/puzzles/#080210. 2010.